

清華大學电子工程系
Department of Electronic Engineering, Tsinghua University
商汤 sensetime

GLITCHES: GPU-FPGA LLM Inference Through a Collaborative Heterogeneous System

Fan Yang*, **Xinhao Yang***, Hongyi Wang, Zehao Wang, Zhenhua Zhu, Shulin Zeng, Yu Wang

Tsinghua University, SenseTime Inc.

xh-yang24@mails.tsinghua.edu.cn, yu-wang@tsinghua.edu.cn

February 28, 2025

NICS EFC

目录 Contents

1 Background

2 Motivation

3 Methods

4 Experiments

目录 Contents

1 Background

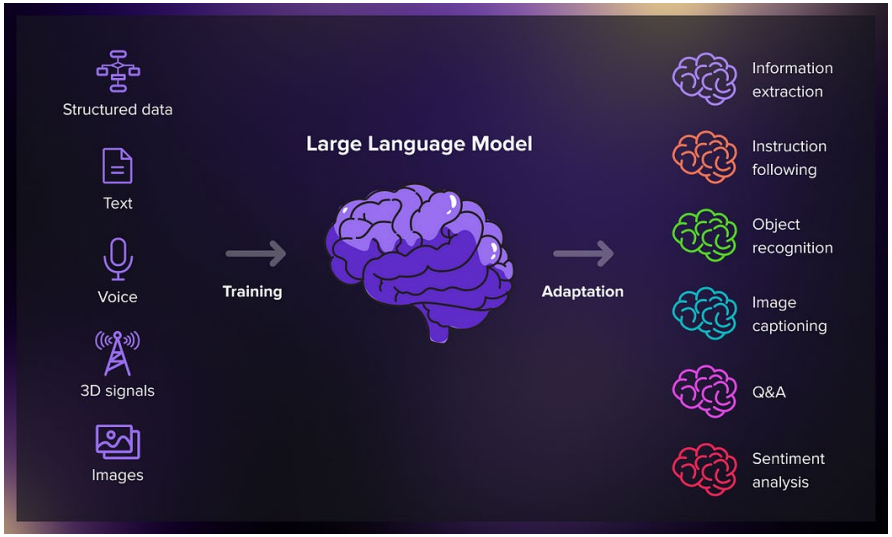
2 Motivation

3 Methods

4 Experiments

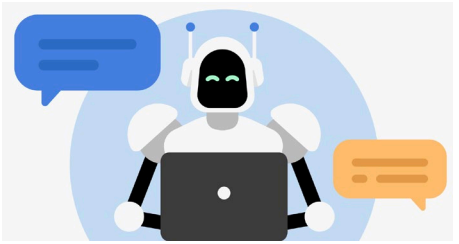
Background: LLMs

- Extensive application scenarios of large language models (LLMs)
  - Already achieved excellent performance in dialogue, information extraction, and so on
  - Many latency-sensitive scenarios favor **batch size = 1**



Large language models (LLMs) adapt to multiple tasks

Code completion



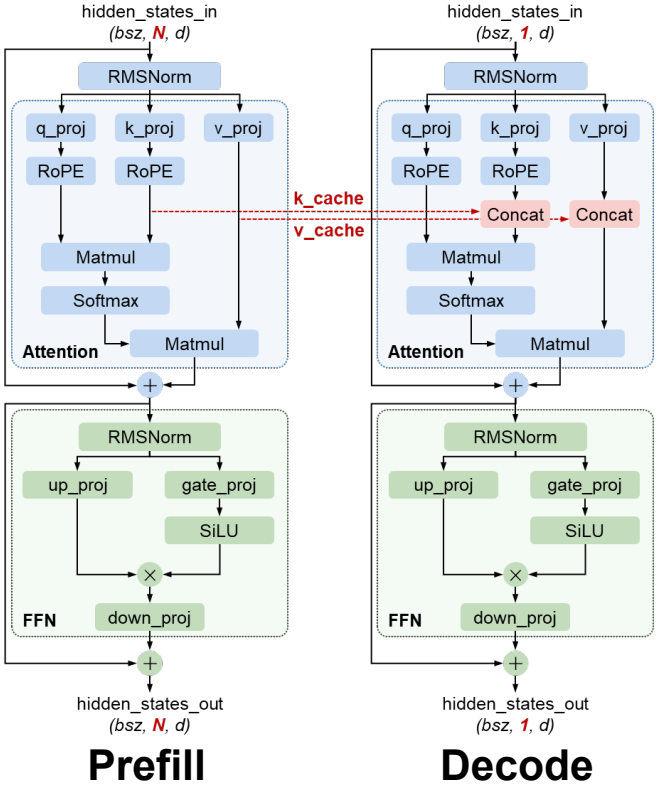
Real-time chatbots

Latency-sensitive scenarios

Background: Prefill and Decode

➢Inference process for large language model

- Prefill and Decode



Prefill stage:

- Comprehend the input prompt and generate the first reply token and **KV cache**
- **Matrix-matrix** multiplications
- Calculate **only once**

Decode stage:

- Generate the next token based on the KV cache **repeatedly**
- **Matrix-vector** multiplications
- Takes about **98%** of the end-to-end time

目录 Contents

1 Background

2 Motivation

3 Methods

4 Experiments

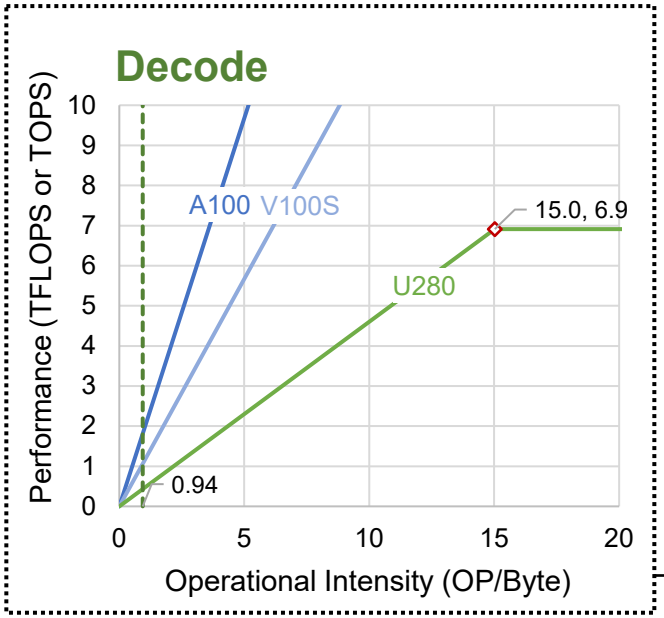
GPU v.s. FPGA: An Example

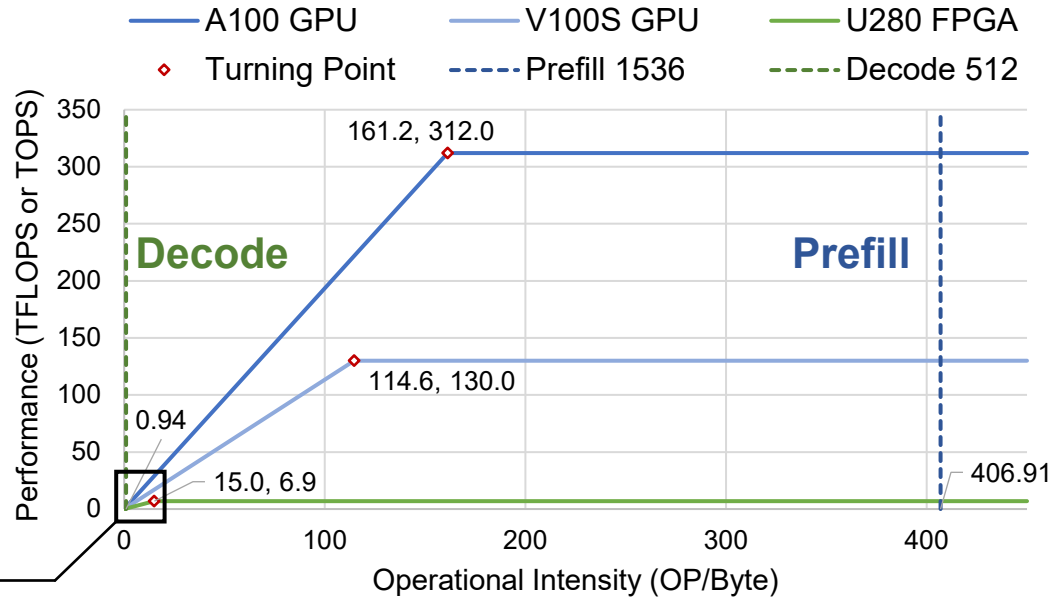
- Llama-2-7B model, [input, output] = [1536, 512]
  - **A100/V100S GPU**: **fast in prefill but high power consumption (~200W)**
  - **U280 FPGA**: **low power consumption (~46W), better decode performance but very poor in prefill**

	Prefill (1536 tokens)					Decode (512 tokens, per token on average)				
Hardware Platform	Computation (GFLOPs or GOPs)	Memory Access (GB)	Operational Intensity (OP/Byte)	Power (W)	Latency (ms)	Computation (GFLOPs or GOPs)	Memory Access (GB)	Operational Intensity (OP/Byte)	Power (W)	Latency (ms)
A100 GPU	21137.01	48.38	406.91	256.6	175.85	14.16	14.08	0.94	167.3	24.26
V100S GPU	21137.01	48.38	406.91	239.8	398.80	14.16	14.08	0.94	222.5	29.52
U280 FPGA	21137.01	21.29	924.47	46.0	5001.20	14.16	3.96	3.33	46.0	21.50

GPU v.s. FPGA: Roofline Model

- Here is the roofline model of prefill and decode in this example:
  - Decode is severely **memory-bound**
  - Prefill is **compute-bound**





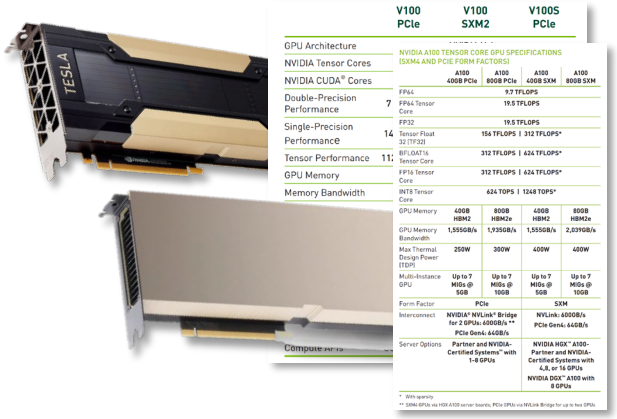
GPU v.s. FPGA: Efficiency

- Llama-2-7B model, [input, output] = [1536, 512]
  - **A100/V100S GPU: Better energy efficiency (token/s/W) and cost efficiency (token/s/$) in prefill** (although with higher price) **but low utilization in decode (<0.4%)**
  - **U280 FPGA: Higher energy efficiency and cost efficiency in decode but low peak performance**

	Parameters			Prefill (1536 tokens)				Decode (512 tokens, per token on average)			
Hardware Platform	Peak Perf. (TFLOPS or TOPS)	Band-width (GB/s)	Cost ($)	Band-width Util.	Compute Util.	Energy Efficiency (token/s/W)	Cost Efficiency (token/s/$)	Band-width Util.	Compute Util.	Energy Efficiency (token/s/W)	Cost Efficiency (token/s/$)
A100 GPU	312[1]	1935	17000	14.22%	38.52%	2.22E-02	3.35E-04	29.99%	0.19%	2.46E-01	2.42E-03
V100S GPU	130[1]	1134	12000	10.70%	40.77%	1.05E-02	2.09E-04	42.06%	0.37%	1.52E-01	2.82E-03
U280 FPGA	6.91[2]	460	8000	0.93%	61.15%	4.35E-03	2.50E-05	40.08%	9.53%	1.01E+00	5.81E-03

[1] Peak computing power of tensor core in GPUs.
[2] The peak INT8 computing power of the U280 FPGA is 24.5 TOPS, but the peak computing power of FlightLLM is only 6.91 TOPS.

GPU v.s. FPGA: Summary



	V100 PCIe	V100 SXM2	V100S PCIe
GPU Architecture	NVIDIA Volta		
NVIDIA Tensor Cores	640		
NVIDIA CUDA® Cores	5,120		
Double-Precision Performance	7 TFLOPS		
Single-Precision Performance	14 TFLOPS		
Tensor Performance	112 TFLOPS		
GPU Memory	32 GB HBM2		
Memory Bandwidth	1134 GB/sec		

NVIDIA A100 TENSOR CORE GPU SPECIFICATIONS (SXM4 AND PCIE FORM FACTORS)

	A100 40GB PCIe	A100 80GB PCIe	A100 40GB SXM	A100 80GB SXM
FP64	9.7 TFLOPS			
FP64 Tensor Core	19.5 TFLOPS			
FP32	19.5 TFLOPS			
Tensor Float 32 (TF32)	156 TFLOPS | 312 TFLOPS*			
BFLOAT16 Tensor Core	312 TFLOPS | 624 TFLOPS*			
FP16 Tensor Core	312 TFLOPS | 624 TFLOPS*			
INT8 Tensor Core	624 TOPS | 1248 TOPS*			
GPU Memory	40GB HBM2	80GB HBM2e	40GB HBM2	80GB HBM2e
GPU Memory Bandwidth	1,555GB/s	1,935GB/s	1,555GB/s	2,039GB/s
Max Thermal Design Power (TDP)	250W	300W	400W	400W
Multi-Instance GPU	Up to 7 MIGs @ 5GB	Up to 7 MIGs @ 10GB	Up to 7 MIGs @ 5GB	Up to 7 MIGs @ 10GB
Form Factor	PCIe		SXM	
Interconnect	NVIDIA® NVLink® Bridge for 2 GPUs: 600GB/s ** PCIe Gen4: 64GB/s		NVLink: 600GB/s PCIe Gen4: 64GB/s	
Server Options	Partner and NVIDIA-Certified Systems™ with 1-8 GPUs		NVIDIA HGX™ A100-Partner and NVIDIA-Certified Systems with 4,8, or 16 GPUs NVIDIA DGX™ A100 with 8 GPUs	

FEATURES	ALVEO U280
Peak INT8 TOPs	24.5
HBM2 Memory Bandwidth	460GB/s
DDR Memory Bandwidth	38GB/s
Internal SRAM Bandwidth	30TB/s
Look-Up Tables	1,079K
PCI Express	Gen4 x8 with CCIX
Thermal Options	Passive or Active*

*Passively cooled cards are production deployable.
Actively cooled cards are for development only.

NVIDIA V100S/A100 GPU

- Memory: 32 / 80 GB HBM2 / 2e
- Bandwidth: 1134 / 2039 GB/s
- Peak Perf*: 130 / 312 TFLOPS
- Cost: 12000 / 17000 $

*Peak perf of tensor core

High bandwidth and performance

➢ **Good at prefill, but low util. in decode**

Xilinx Alveo U280 FPGA

- Memory: 8 GB HBM2 + 32 GB DDR
- Bandwidth: 460 GB/s + 38 GB/s
- Peak Perf: 24.5 TOPS (INT8)
- Cost: 8000 $

Not bad bandwidth but low performance

➢ **Good at decode, but slow in prefill**

Motivation

How to combine the strength of both GPUs and FPGAs?

Heterogeneous!

GPUs for Prefill, FPGAs for Decode

目录 Contents

1 Background

2 Motivation

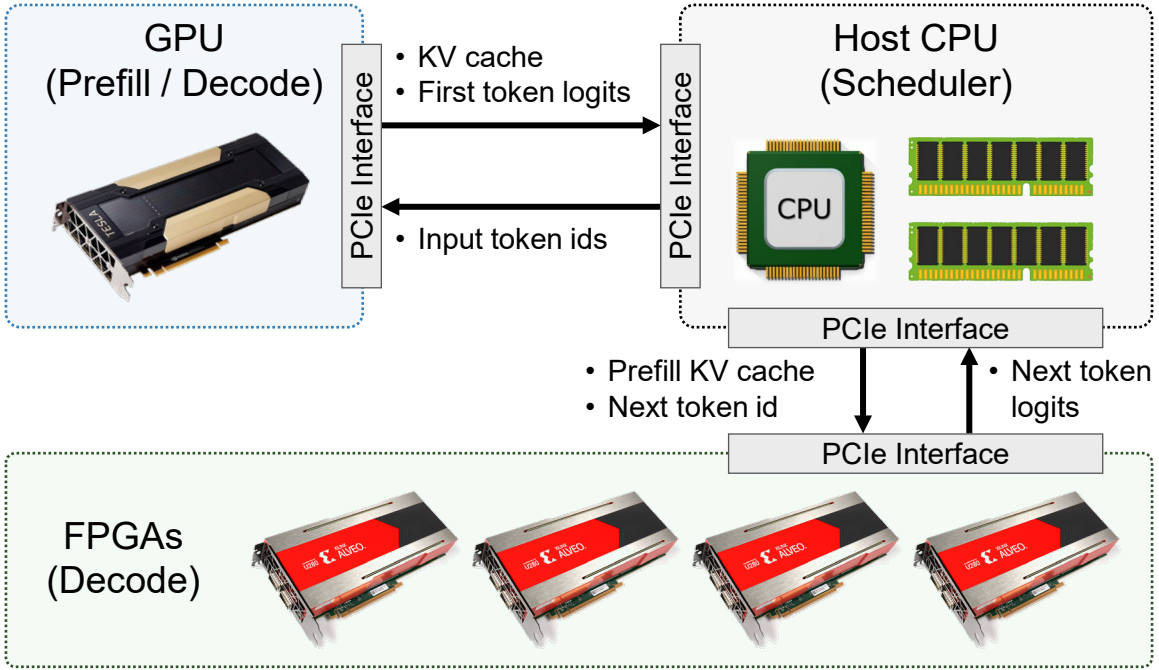
3 Methods

4 Experiments

Overall Architecture of GLITCHES

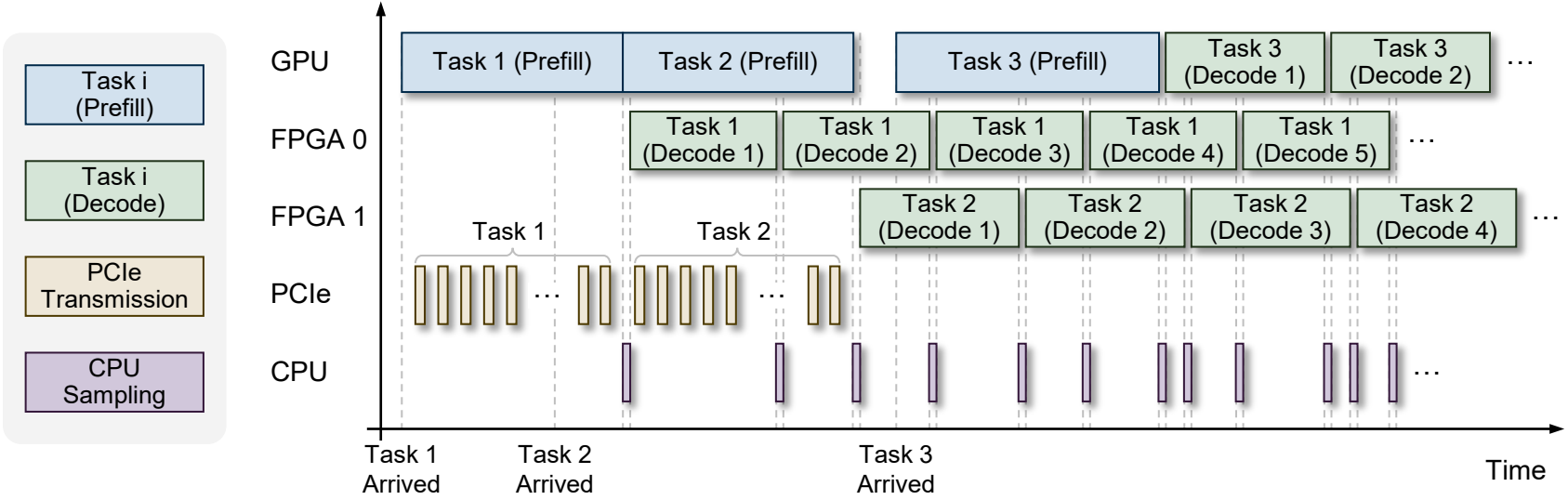
➢GLITCHES: A GPU + FPGA Heterogeneous System

- Host CPU:
  - Task scheduling
- GPUs:
  - Prefill tasks
  - Decode tasks when FPGAs are all busy
- FPGAs:
  - Decode tasks
- Data transmission
  - Via PCIe interface



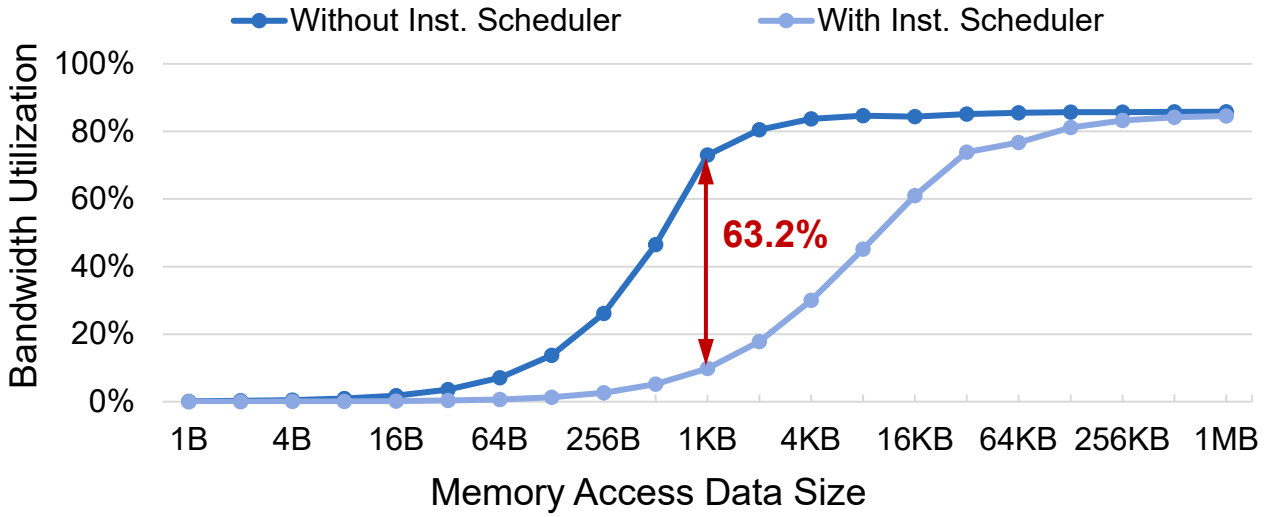
Timeline: How GLITCHES Works

- When a inference task arrives:
  - Distribute the prefill task to the GPUs
  - Transmit KV cache from GPU to the FPGA via PCIe during prefill computation
  - Allocate decode tasks to an idle FPGA
    - When all FPGAs are busy, GPUs assist with the decode task



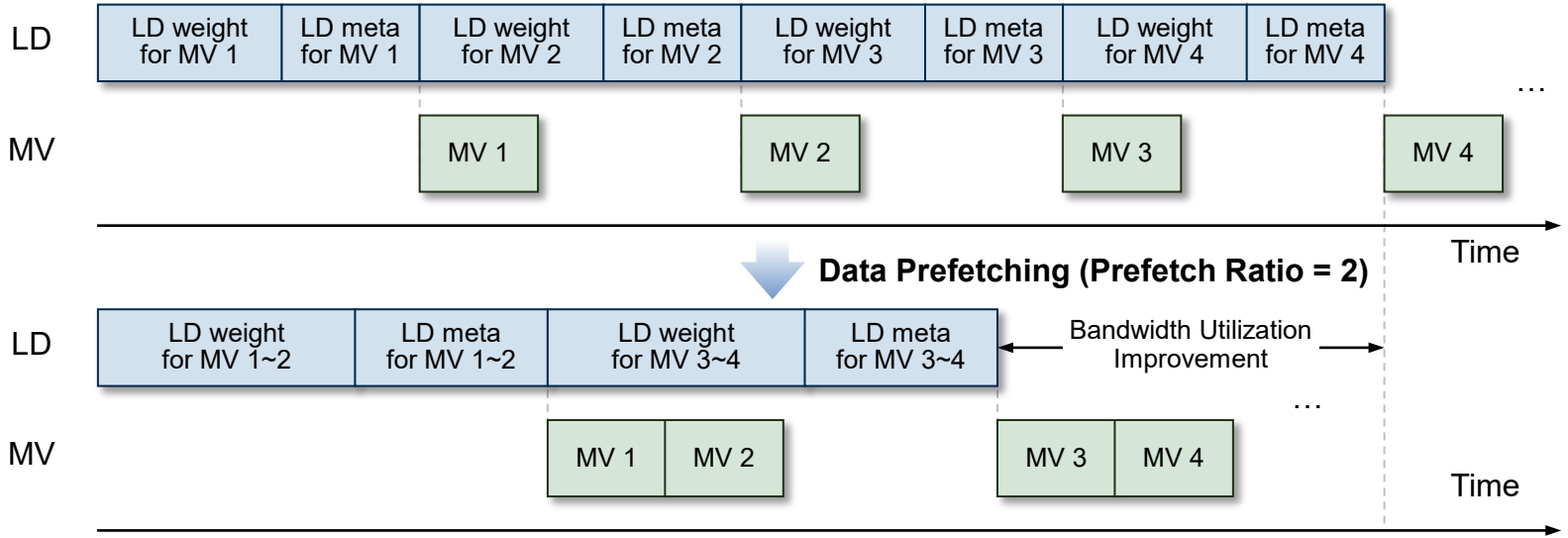
Data Prefetching: For Faster FPGA Decode

- For better decode performance on FPGAs:
  - Instruction scheduling overhead has a significant impact on **fine-grained memory access (256B ~ 16KB)** performance **(up to 63.2%)**
  - Merging multiple load requests with data prefetching can improve the memory-bound decode stage performance



Data Prefetching: For Faster FPGA Decode

- For better decode performance on FPGAs:
  - Instruction scheduling overhead has a significant impact on fine-grained memory access (256B ~ 16KB) performance (up to 63.2%)
  - **Merging multiple load requests with data prefetching** (from [256B ~ 8KB] to [1KB ~ 32KB]) improves the memory-bound decode stage performance



目录 Contents

1 Background

2 Motivation

3 Methods

4 Experiments

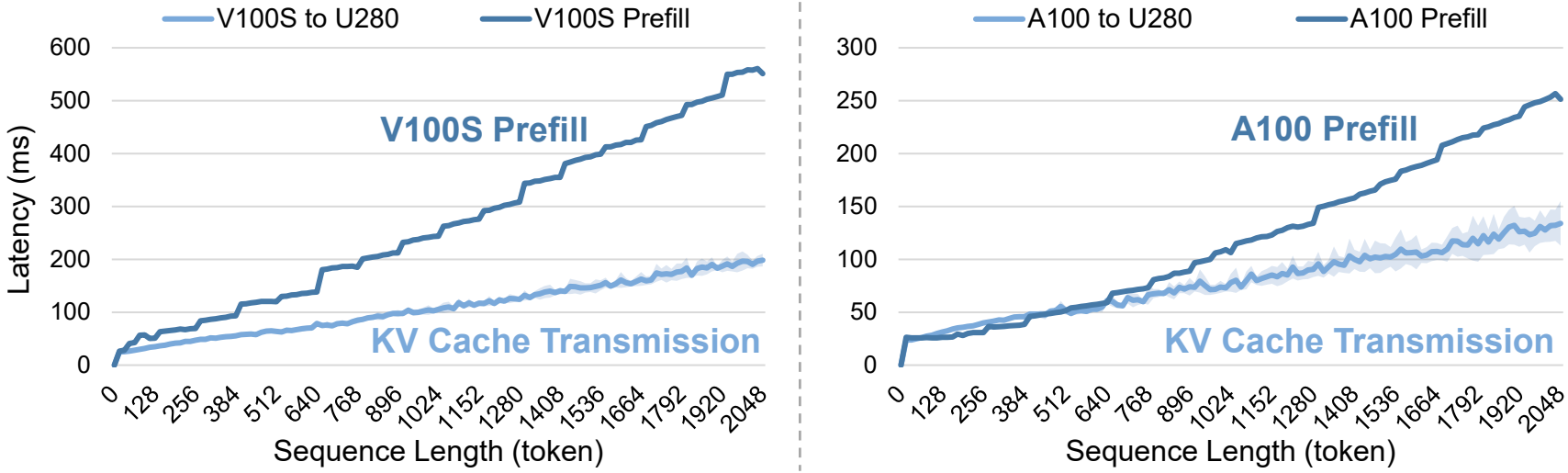
Experiment Setup

- Hardware Platform (8-card within a single node):
  - **GPU-only:**
    - 8 * NVIDIA V100S/A100
  - **FPGA-only:**
    - 8 * Xilinx Alveo U280
  - **GLITCHES: 1 GPU + 7 FPGA**
    - 1 * NVIDIA V100S/A100 + 7 * Xilinx Alveo U280
- Framework:
  - GPU: HuggingFace (FP16)
  - FPGA: FlightLLM (about W4A8) [1]
- Model:
  - Llama-2-7B (about 12.6GB weights in FP16, 3.6GB weights in W4)

[1] Zeng, Shulin, et al. "Flightllm: Efficient large language model inference with a complete mapping flow on fpgas." *Proceedings of the 2024 ACM/SIGDA International Symposium on Field Programmable Gate Arrays*. 2024.

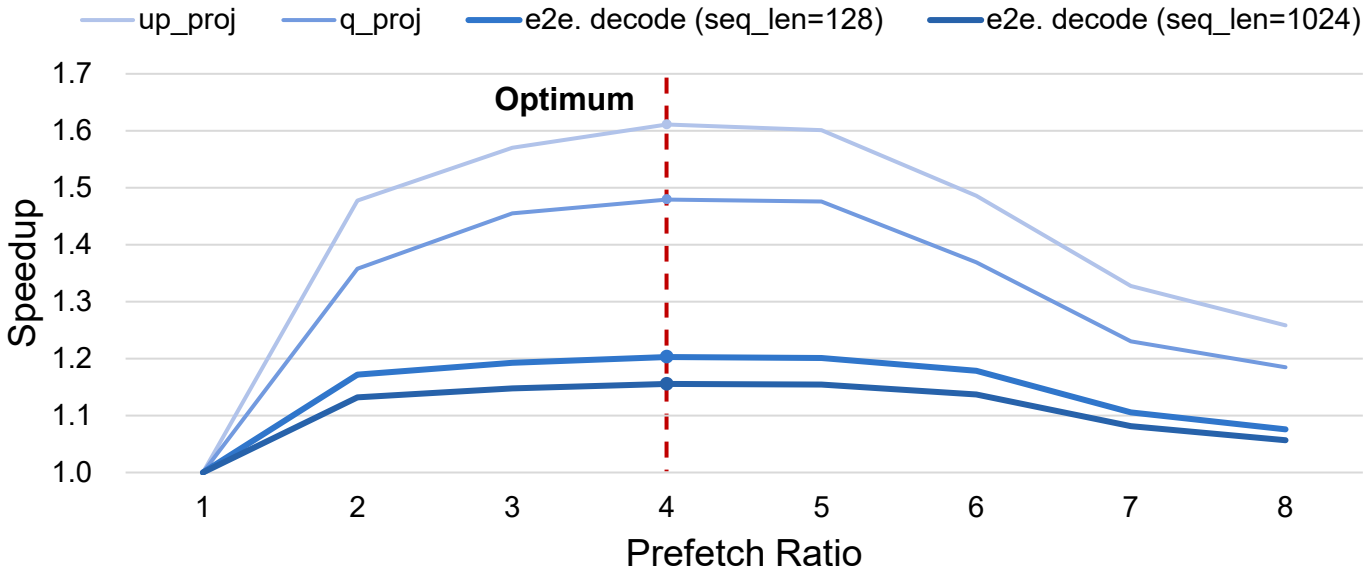
KV Cache Transmission and Prefill Latency

- KV cache transmission latency and prefill latency for different tokens
  - KV cache transmission latency from GPU memory to HBM on FPGA is less than the GPU prefill latency (in most cases)
    - KV cache transmission can be **overlapped by prefill computation**
  - Prefill computation grows **quadratically** with the sequence length



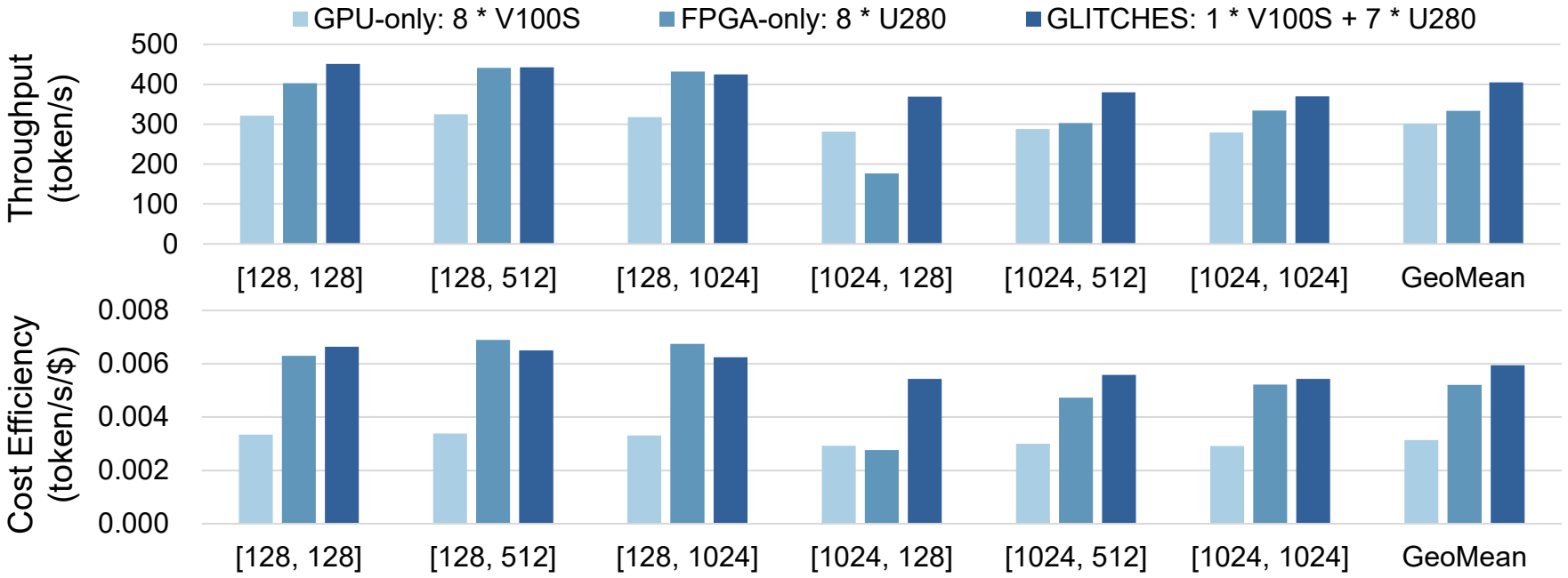
Data prefetching for FPGA decode

- Merging multiple load requests with data prefetching
  - The optimal prefetch ratio is **4**
  - Improved the end-to-end decode performance by **1.20x and 1.16x** when the sequence length is 128 and 1024, respectively



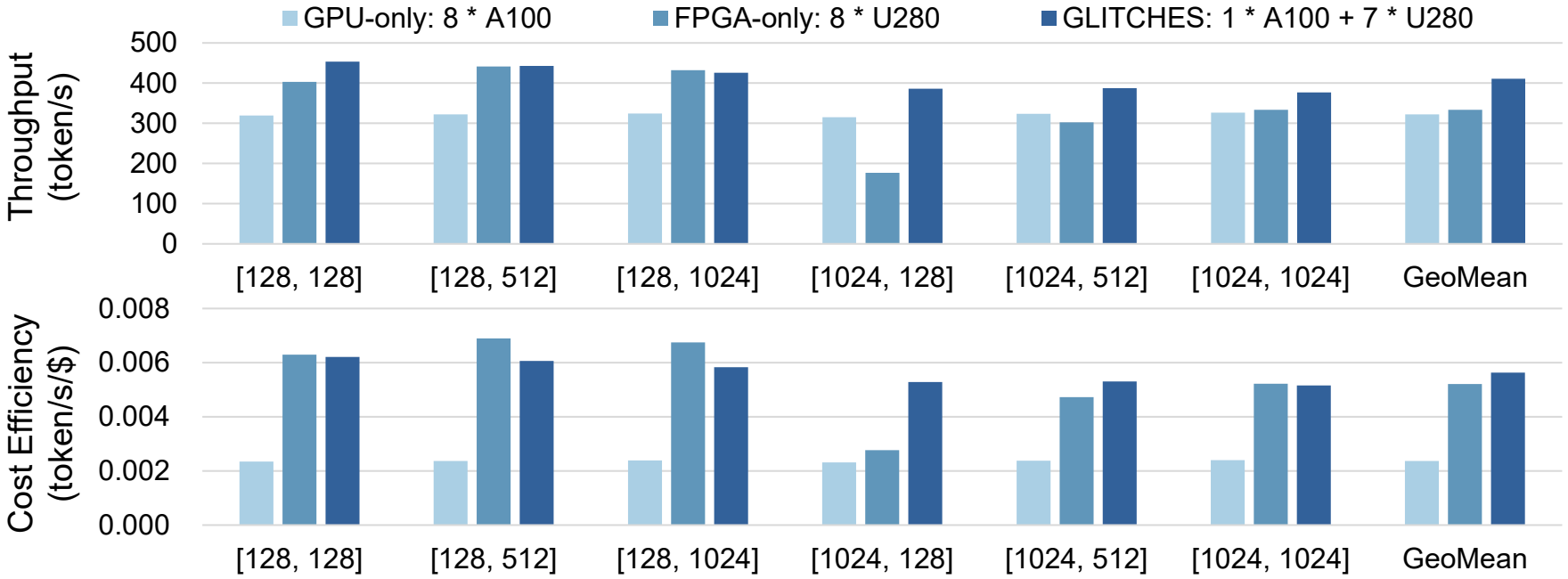
End-to-end Inference: to V100S GPU

- For GLITCHES with **one V100S GPU and seven U280 FPGAs**:
  - Achieves an average throughput (token/s) of **1.34/1.21x** and cost efficiency (token/s/$) of **1.90/1.14x** compared to an 8-card V100S/U280-only system



End-to-end Inference: to A100 GPU

- For GLITCHES with **one A100 GPU and seven U280 FPGAs**:
  - Improves the average throughput (token/s) by **1.28/1.23x** and cost efficiency (token/s/$) by **2.38/1.08** compared to an 8-card A100/U280-only system



Thanks and Q&A

Fan Yang*, **Xinhao Yang***, Hongyi Wang, Zehao Wang, Zhenhua Zhu, Shulin Zeng, Yu Wang

Tsinghua University, SenseTime Inc.

xh-yang24@mails.tsinghua.edu.cn, yu-wang@tsinghua.edu.cn

February 28, 2025